TIPE

Comment simuler une évacuation d'un lieu clos : étude de flux de personnes.

Annexes

Sommaire :

I. Code Python

a. Fond

```python
# -*- coding: utf-8 -*-
"""

TIPE LAGNEAU CLEMENT 2019

Creation des fond

"""

"""
Sur les fonds :
    0 : libre
    -1 : mur
"""

def fond(n):
    """
    Cree un fond avec des murs sur les cotes
    """
    t=[[0 for x in range(n)] for x in range(n)]
    for x in range(n):
        t[0][x]=2
        t[x][0]=2
        t[n-1][x]=2
        t[x][n-1]=2
    return(t)

def fond_1(n):
    """
    Cree un fond avec des 1 sur les bords et une sortie au centre
    """
    a=fond(n)
    a[n//2][n//2]=-1
    for x in range(1,n-1):
        a[1][x]=1
        a[x][1]=1
        a[n-2][x]=1
        a[x][n-2]=1
    return(a)

def fond_2(n):
    """
    Cree un fond avec des 1 dans les coins et une sortie dans le dernier coin
    """
    a=fond(n)
    a[n-2][n-2]=-1
    a[1][1]=1
    a[1][n-2]=1
    a[n-2][1]=1
    return(a)
```

```python
def fond_3(n):
    """
    Cree un fond avec des 1 dans les coins et une sortie dans le dernier coin
    avec un mur
    """
    a=fond(n)
    a[n-2][n-2]=-1
    a[1][1]=1
    a[1][n-2]=1
    a[n-2][1]=1
    for x in range(n//2):
        a[x][n//2-1]=2
    return(a)


def avion(n):
    """
    Cree un fond de type avion
    """
    a=[[0 for x in range(n)]for x in range(n)]
    for x in range(n):
        a[0][x]=2
        a[x][0]=2
        a[n-1][x]=2
        a[x][n-1]=2
    for x in range(n//2):
        for y in range(1,n-1):
            a[2*x][y]=2
            a[2*x+1][y]=1
            a[n-1][y]=2
    for x in range(1,n-1):
        a[x][n//2]=0
        a[x][n//2-1]=0
    a[n-1][n//2]=-1
    a[n-1][n//2-1]=-1
#    a[0][n//2]=-1
#    a[0][n//2-1]=-1
    return(a)

```

b. Itération

```python
# -*- coding: utf-8 -*-
"""

TIPE CLEMENT LAGNEAU 2019

Iteration
"""

"""
-1 : sortie
0 : libre
1 : il y a quelqu un
2 : mur
"""


from copy import deepcopy

def avance(a,b,score):
    """
    Renvoie le tableau et le score avance avec une iteration
    """
    n=len(a)
    c=deepcopy(a)
    for ligne in range(n):
        for colonne in range(n):
            if a[ligne][colonne]==1:
                # On avance
                if b[ligne][colonne]=="d":
                    if a[ligne][colonne+1]==0 and c[ligne][colonne+1]==0:
                        c[ligne][colonne]=0
                        c[ligne][colonne+1]=1
                    if a[ligne][colonne+1]==-1:
                        c[ligne][colonne]=0
                        score+=1
                if b[ligne][colonne]=="g":
                    if a[ligne][colonne-1]==0 and c[ligne][colonne-1]==0:
                        c[ligne][colonne]=0
                        c[ligne][colonne-1]=1
                    if a[ligne][colonne-1]==-1:
                        c[ligne][colonne]=0
                        score+=1
                if b[ligne][colonne]=="b":
                    if a[ligne+1][colonne]==0 and c[ligne+1][colonne]==0:
                        c[ligne][colonne]=0
                        c[ligne+1][colonne]=1
                    if a[ligne+1][colonne]==-1:
                        c[ligne][colonne]=0
                        score+=1
                if b[ligne][colonne]=="h":
                    if a[ligne-1][colonne]==0 and c[ligne-1][colonne]==0:
                        c[ligne][colonne]=0
                        c[ligne-1][colonne]=1
                    if a[ligne-1][colonne]==-1:
                        c[ligne][colonne]=0
                        score+=1
    return(c,score)

```

c. Stratégie

```
# -*- coding: utf-8 -*-
"""

TIPE CLEMENT LAGNEAU

Strategie
"""



"""
A[ligne][colonne]
strategie :
    h : haut
    b : bas
    d : droite
    g : gauche
    i : impossible
"""

import random

def strategie(t):
    """
    Renvoie une strategie adaptee au fond t
    """
    n=len(t)
    strat=[["" for x in range(n)] for x in range(n)]
    for x in range(n):
        for y in range(n):
            if t[x][y] == 2:
                strat[x][y]="i"
            elif t[x][y] == -1:
                strat[x][y]="s"
            else:
                r=random.random()
                if r<=0.25:
                    strat[x][y]="h"
                elif 0.25<r<=0.5:
                     strat[x][y]="b"
                elif 0.5<r<=0.75:
                     strat[x][y]="d"
                else:
                     strat[x][y]="g"
    return(strat)

def strategie_case(t,x,y):
    """
    Renvoie une strategie adaptee a la case x y du fond t
    """
    if t[x][y] == "i":
        return("i")
    elif t[x][y] == "s":
        return("s")
    else:
        r=random.random()
        if r<=0.25:
            return("h")
        elif 0.25<r<=0.5:
            return("b")
        elif 0.5<r<=0.75:
             return("d")
        else:
             return("g")

```

d. Dessiner

```
# -*- coding: utf-8 -*-
"""

TIPE CLEMENT LAGNEAU 2019

Dessiner
"""

"""
-1 : sortie
0 : libre
1 : il y a quelqu un
2 : mur
"""


import tkinter
import iteration
from copy import deepcopy
import fond
import strategie

k=50

#a=fond.fond_test1(n)
#b=strategie.strategie(a)

dico = {-1 : "red", 2 : "yellow",1 :"blue", 0: "white"}

score = 0

def init():
    for x in range(n):
        for y in range(n):
            couleur[y][x]=canvas.create_rectangle((x*k, y*k,(x+1)*k, (y+1)*k), outline="gray", fill="red")

def calcul():
    global score
    global a
    c,score=iteration.avance(a,b,score)
    for x in range(n):
        for y in range(n):
            canvas.itemconfig(couleur[x][y], fill=dico[c[x][y]])
    a=deepcopy(c)

def final():
    calcul()
    fenetre.after(100, final)

couleur = [[0 for x in range(n)] for y in range(n)]

fenetre = tkinter.Tk()
canvas = tkinter.Canvas(fenetre, width=k*n, height=k*n, highlightthickness=0)
canvas.pack()
init()

for x in range(n):
    for y in range(n):
        if b[x][y] == "h":
            canvas.create_line(((y+0.5)*k, (x+0.5)*k, (y+0.5)*k, (x-0.5)*k),arrow='last')
        if b[x][y] == "b":
            canvas.create_line(((y+0.5)*k, (x+0.5)*k, (y+0.5)*k, (x+1.5)*k),arrow='last')
        if b[x][y] == "d":
            canvas.create_line(((y+0.5)*k, (x+0.5)*k, (y+1.5)*k, (x+0.5)*k),arrow='last')
        if b[x][y] == "g":
            canvas.create_line(((y+0.5)*k, (x+0.5)*k, (y-0.5)*k, (x+0.5)*k),arrow='last')

final()
fenetre.mainloop()

```

e. Algorithme génétique

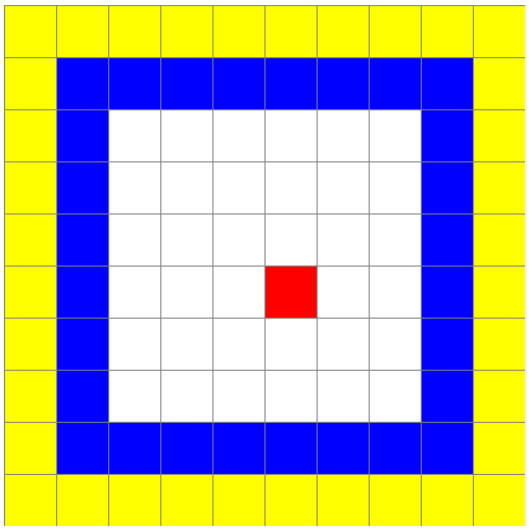
```python
# -*- coding: utf-8 -*-
"""

TIPE CLEMENT LAGNEAU 2019

Algorithme genetique
"""

#Nombre
n= 10

#Rapport visible/n
k = 50

#taille de liste
i=30

it=n*n

import os
import random
import fond
import iteration
import strategie
from copy import deepcopy

def cle(x):
    return(x[0:2])

def nbvivant(t,n):
    res=0
    for i in range(n):
        for j in range(n):
            if t[i][j]==1:
                res += 1
    return(res)

fd=fond.fond_2

vivants = nbvivant(fd(n),n)

# nb de changement dynamique
changement = 10
```

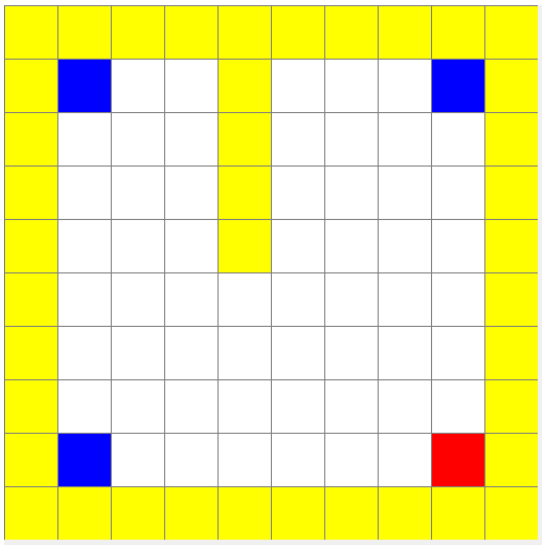
```python
def meilleur_init():
    """
    Renvoie la liste initiale avec comme liste de sortie :
        [0]: score
        [1]: nbiteration max
        [2]: la table de strategie
    """
    liste = []
    for x in range(i):
        a = fd(n)
        b = strategie.strategie(a)
        c = principal(it,a,b)
        liste.append((c[1],c[0],b))
    liste=sorted(liste,key=cle,reverse=True)
    return(liste)

def fusion_liste(liste,n):
    """
    Renvoie la liste apres la mutation
    """
    l=[]
    for x in range(3*i//4):
        l.append(deepcopy(liste[x][2]))
    choix=random.sample(liste,i//8)
    for x in range(i//8):
        l.append(fusion(choix[x][2],n))
    for x in range(i//8):
        l.append(fusion2(liste[x][2],liste[i//8+x][2],n))
    return(l)

def meilleur_apres(liste):
    """
    Renvoie la liste apres une iteration de l algorithme genetique
    """
    liste_apres=[]
    for b in liste:
        a = fd(n)
        c = principal(it,a,b)
        liste_apres.append((c[1],c[0],b))
    liste_apres=sorted(liste_apres,key=cle,reverse=True)
    return(liste_apres)

def principal(nbiteration,a,b):
    """
    Renvoie le nombre max d iteration et le score pour une table
    """
    c = deepcopy(a)
    score=0
    i=0
    while i<nbiteration and score<vivants:
        c,score=iteration.avance(c,b,score)
        i+=1
    return(i,score)

```

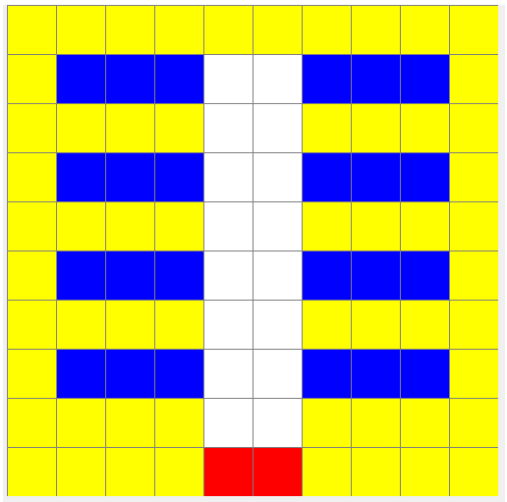
```
def fusion(liste,n):
    """
    Renvoie la liste en effectuant /changement/ changements sur les cases
    """
    r=deepcopy(liste)
    for k in range(changement):
        x=random.randint(0,n-1)
        y=random.randint(0,n-1)
        r[x][y]=strategie.strategie_case(liste,x,y)
    return(r)

def fusion2(liste1,liste2,n):
    """
    Renvoie la liste fusionne de l1 et l2 suivant une proba 1/2
    """
    r=deepcopy(liste1)
    for x in range(10):
        for y in range(10):
            t=random.random()
            if t<0.5:
                r[x][y]=liste2[x][y]
    return(r)


en_cours=[]
lapres=meilleur_init()
indice=0

nom="Res_"+str(n)+"_fond_2_"+str(i)+"_1"


os.chdir("D:/Cours/MPBIS/TIPE/")
res = open(nom,"w")
res.close()

while lapres[0][0]!=vivants and indice<50000:
    res = open(nom,"a")
    res.writelines(str((indice,lapres[0][0],changement))+"\n")
    res.close()
    changement = vivants-lapres[0][0]
    print(indice,lapres[0][0],changement)
    en_cours=fusion_liste(lapres,n)
    lapres=meilleur_apres(en_cours)
    indice +=1

res = open(nom,"a")
res.writelines(str(lapres[0][2]))
res.close()


print(lapres[0][2])

```

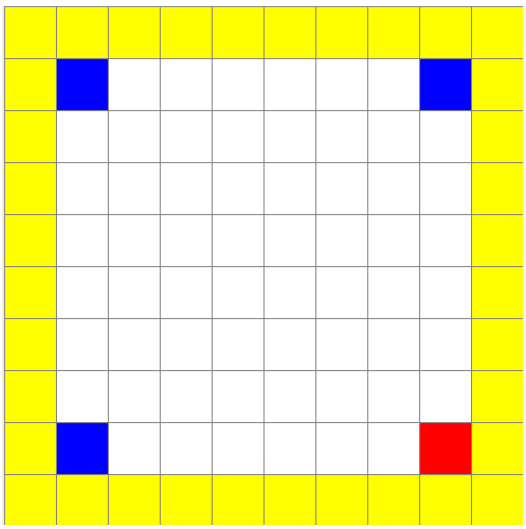
II. Représentation

a. Dessin des fonds

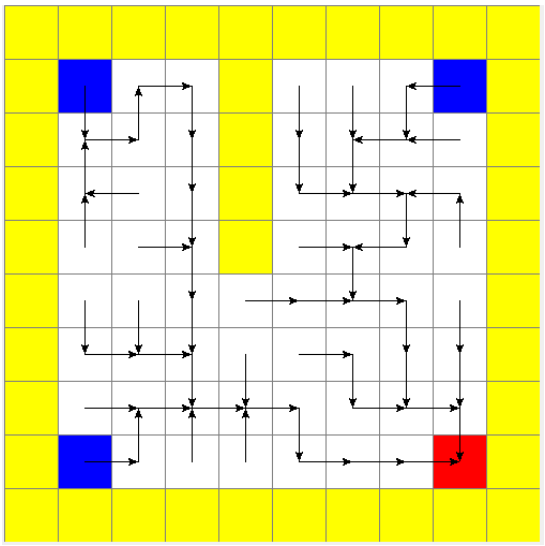


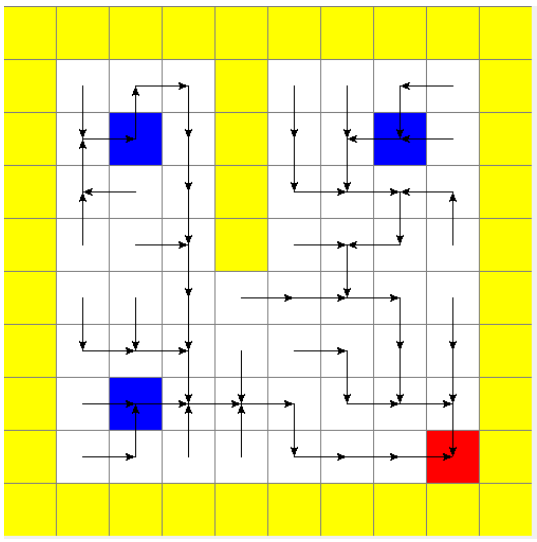


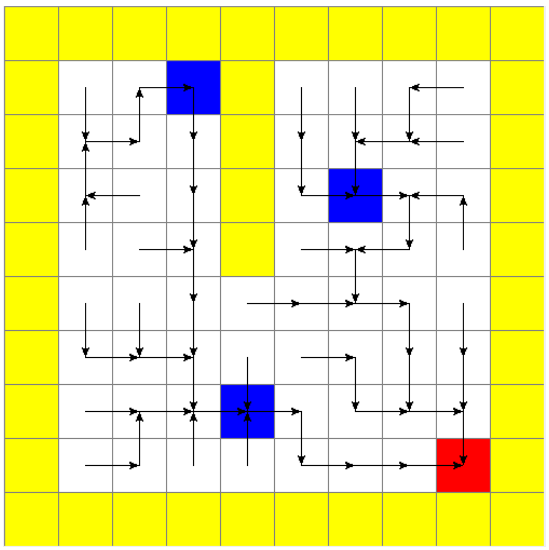


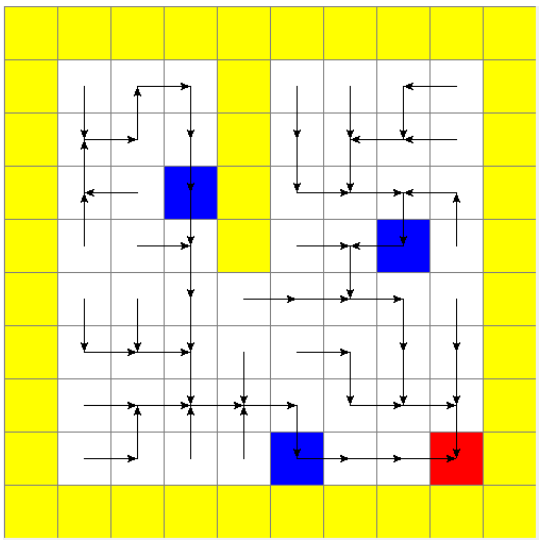


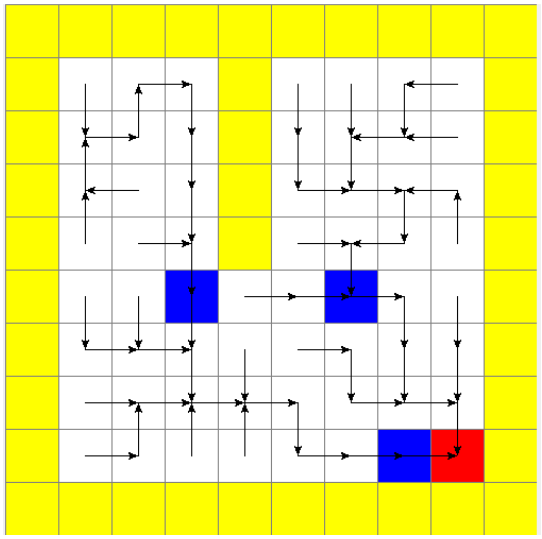
b. Illustration d'une solution

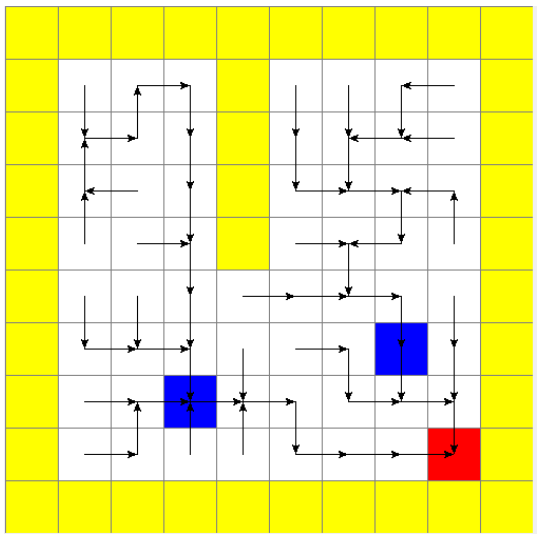


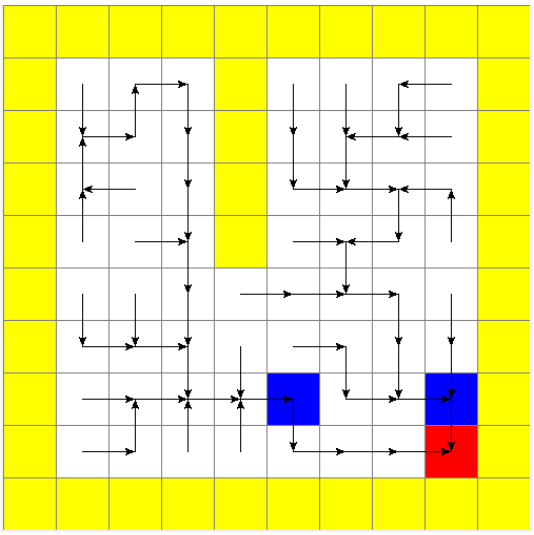


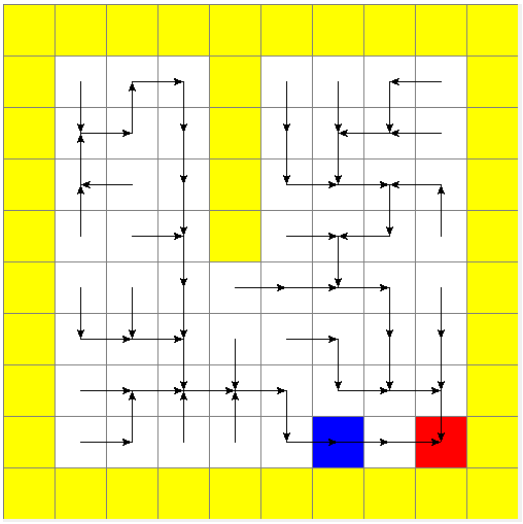


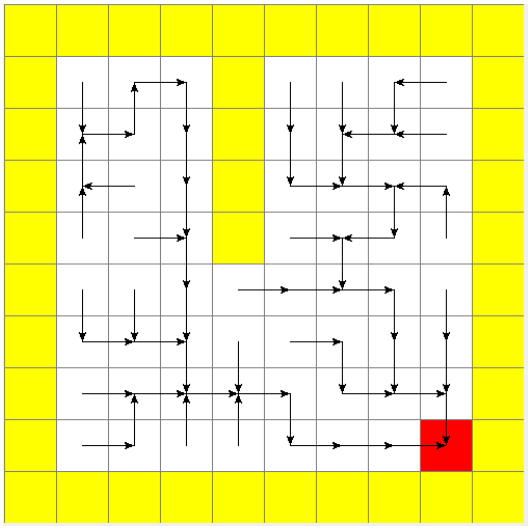




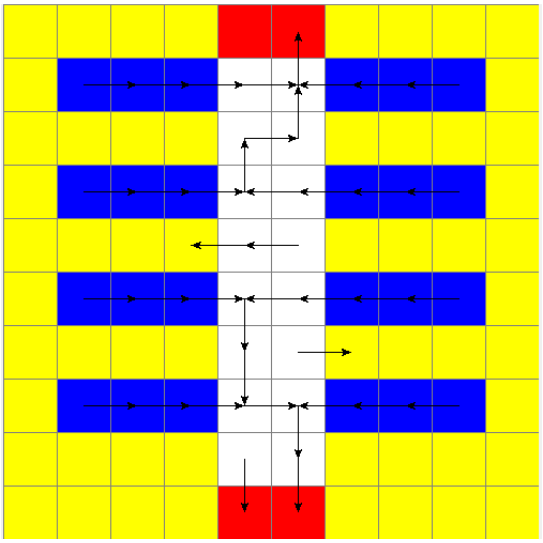


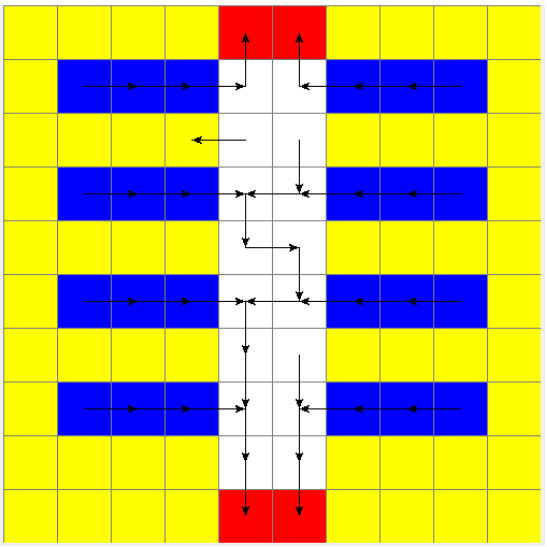


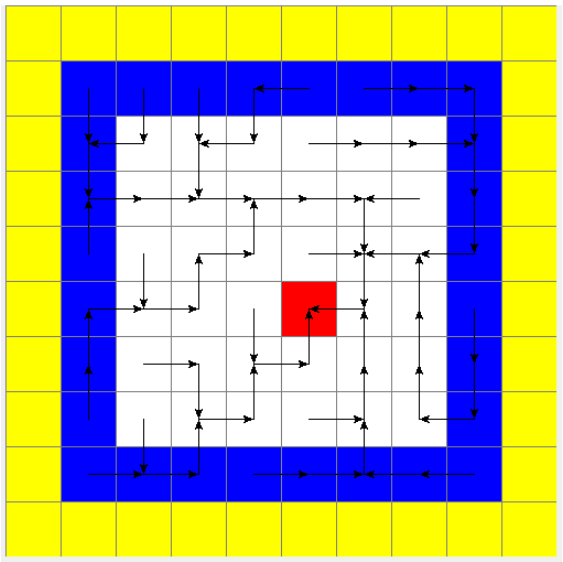


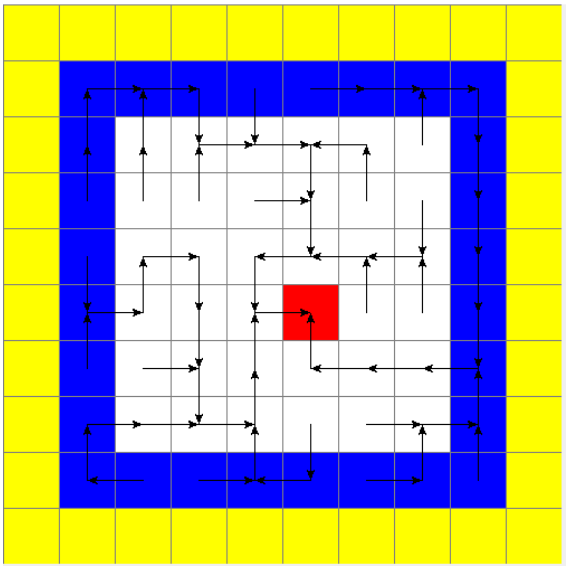


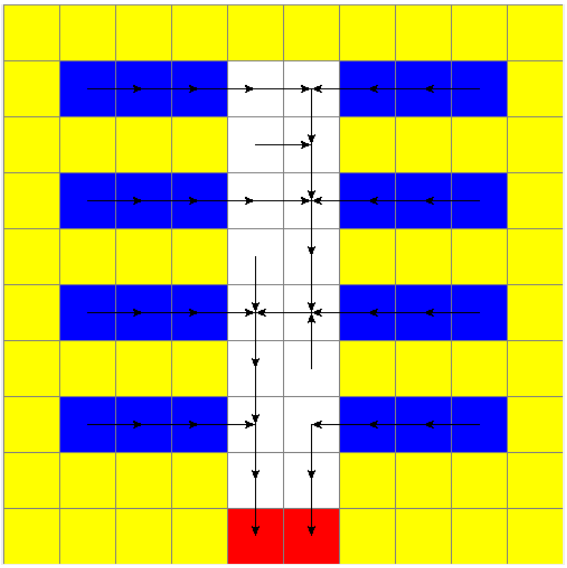
c. Quelques solutions

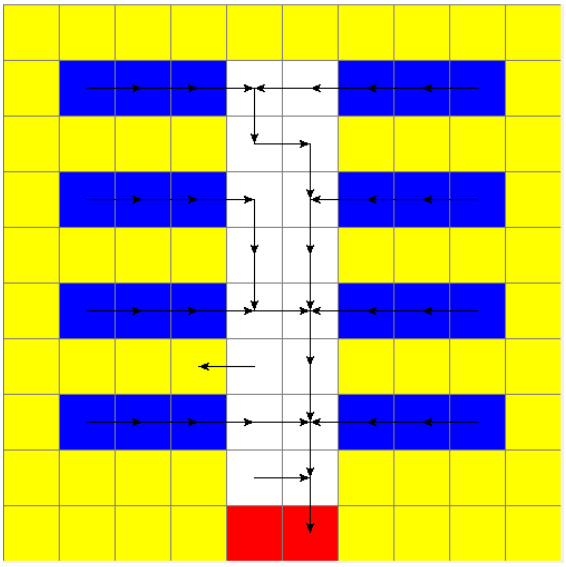


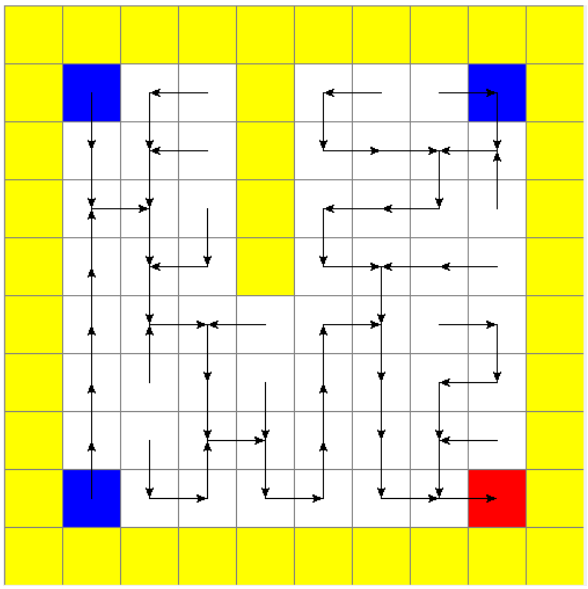


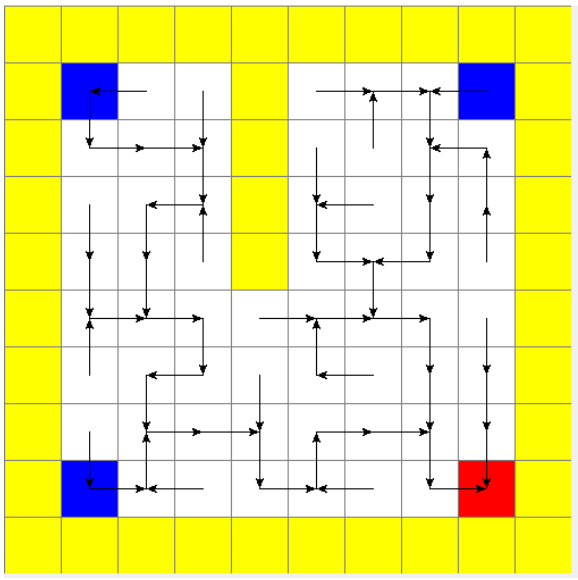


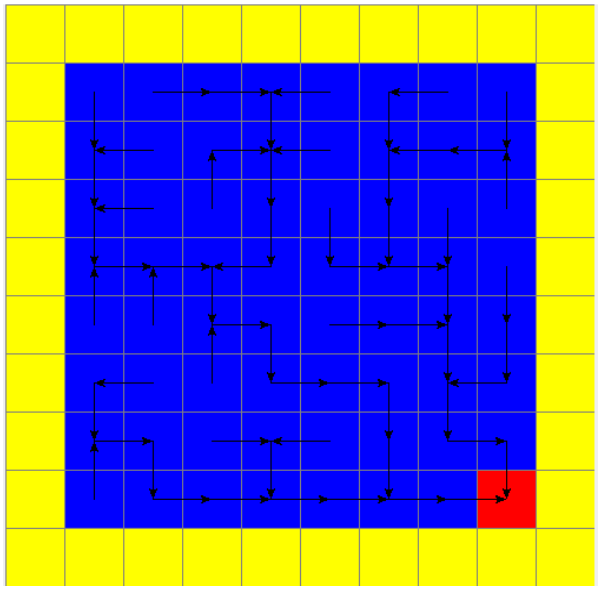


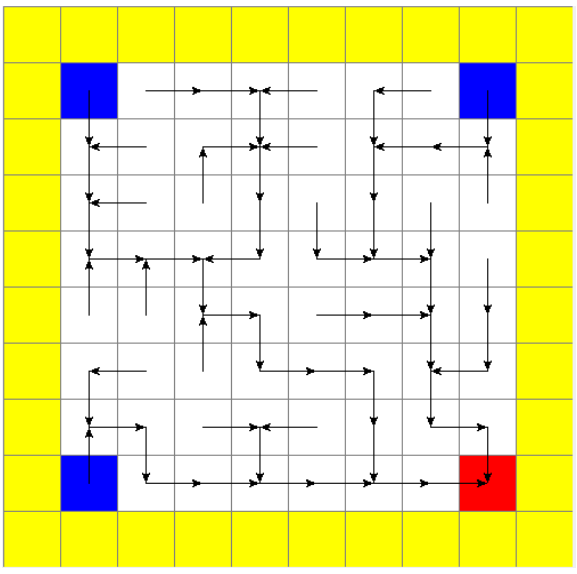


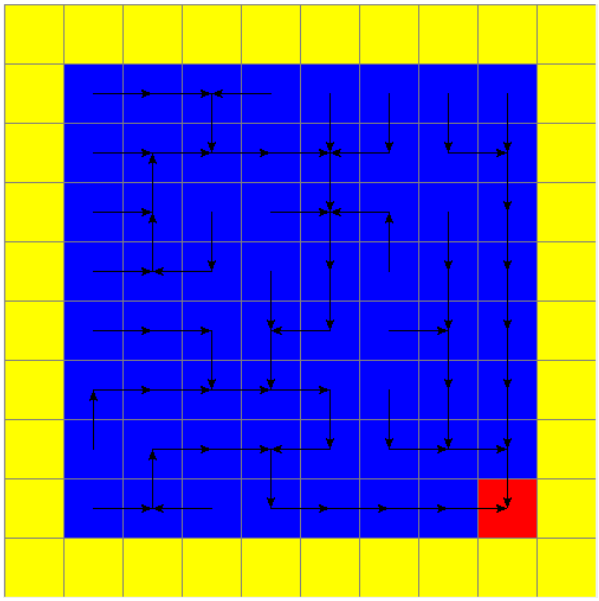


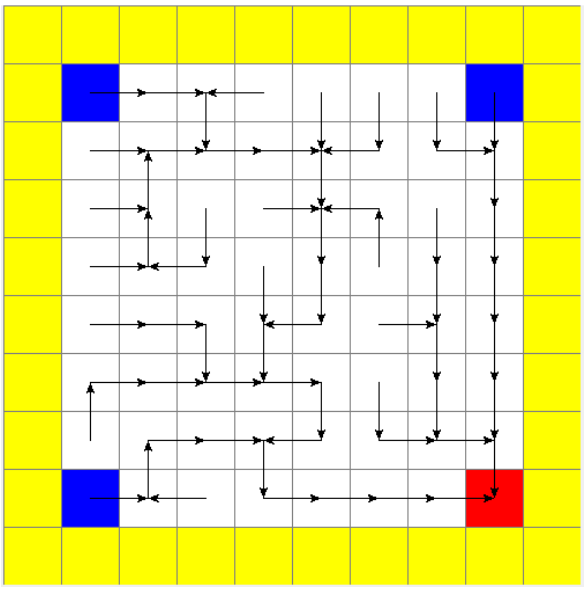






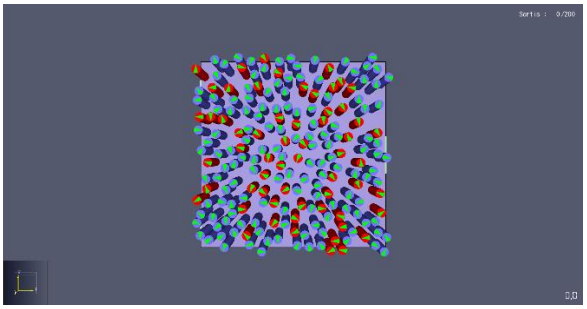




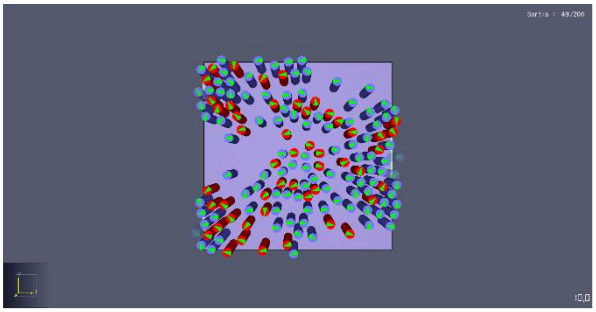


III. Pathfinder

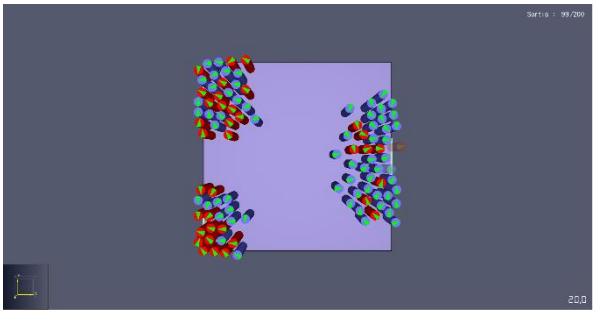
a. Sortie aléatoire (Porte par laquelle nous sommes entrés)



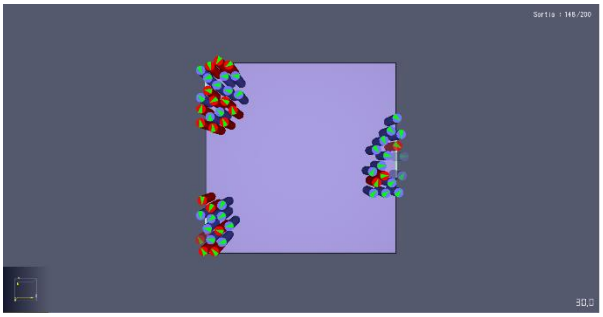
Temps : 0 | Sortis : 0/200



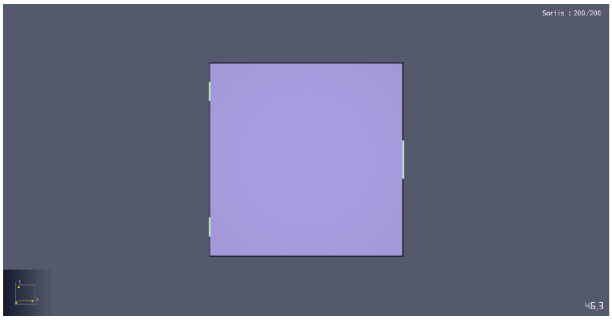
Temps : 10 | Sortis : 49/200



Temps : 20 | Sortis : 99/200

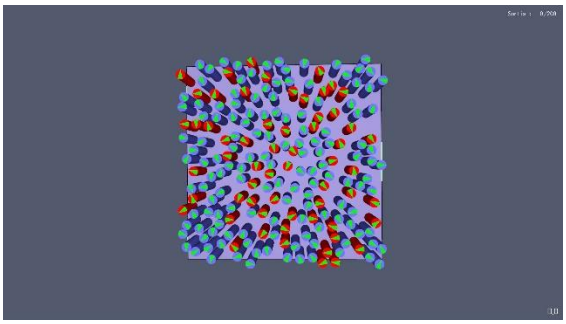


Temps : 30 | Sortis : 148/200

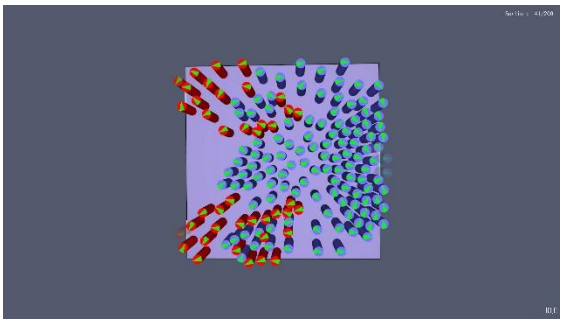


Temps : 46.3 | Sortis : 200/200

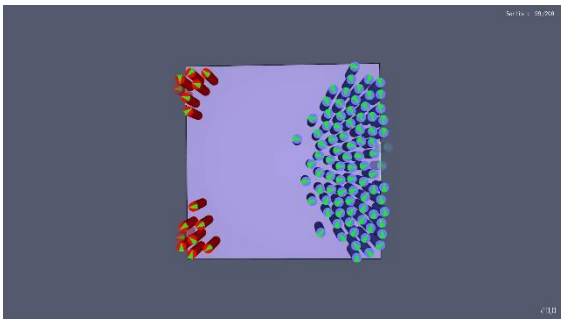
b. Sortie par la porte par laquelle nous sommes entrés



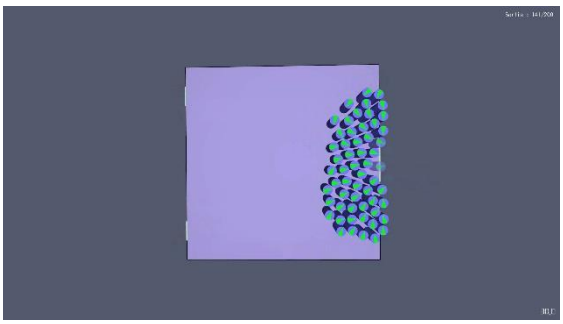
Temps : 0 | Sortis : 0/200



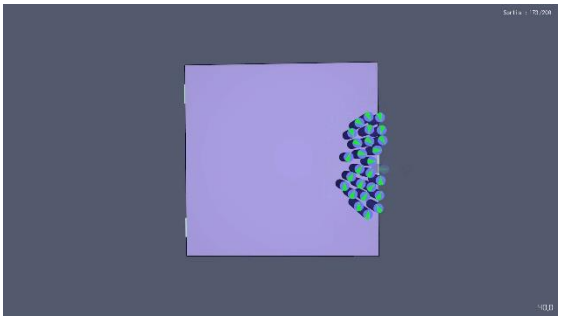
Temps : 10 | Sortis : 41/200



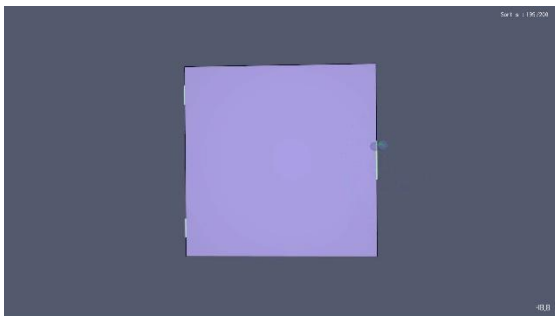
Temps : 20 | Sortis : 99/200



Temps : 30 | Sortis : 141/200

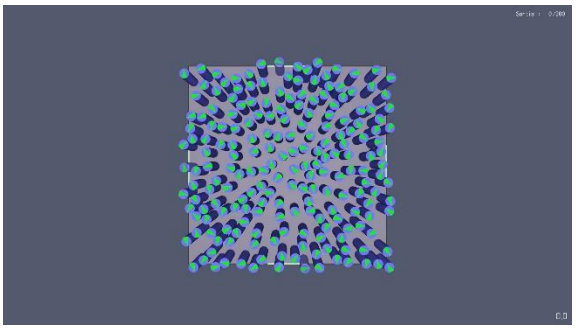


Temps : 40 | Sortis : 173/200

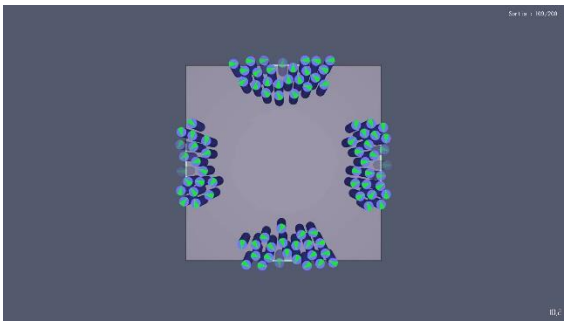


Temps : 48.8 | Sortis : 200/200

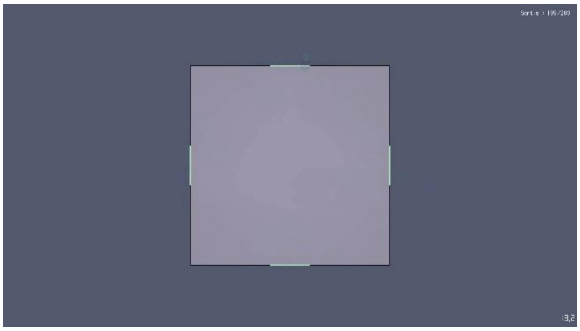
c. 4 sorties



Temps : 0 | Sortis : 0/200

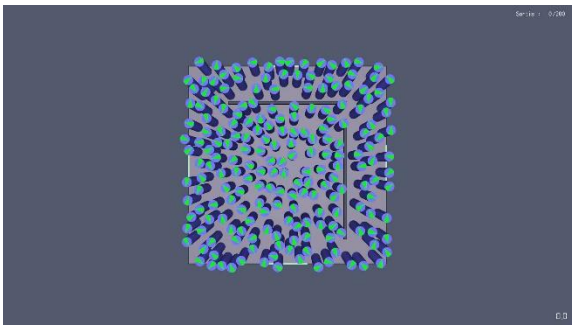


Temps : 10 | Sortis : 109/200

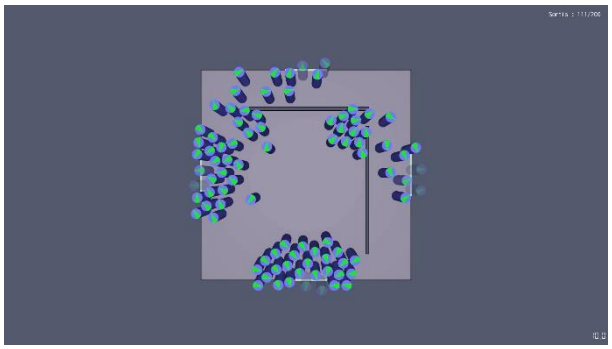


Temps : 19.2 | Sortis : 200/200

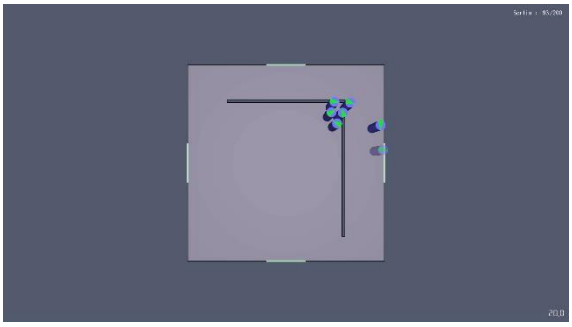
d. 4 sorties avec 2 murs



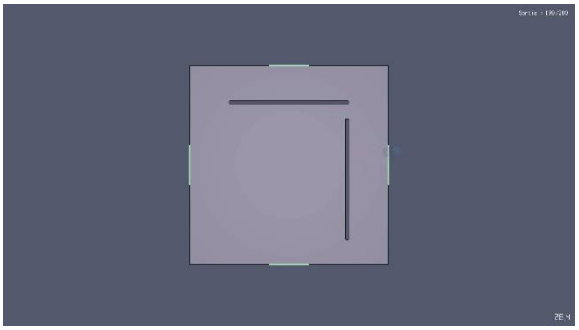
Temps : 0 | Sortis : 0/200



Temps : 10 | Sortis : 111/200

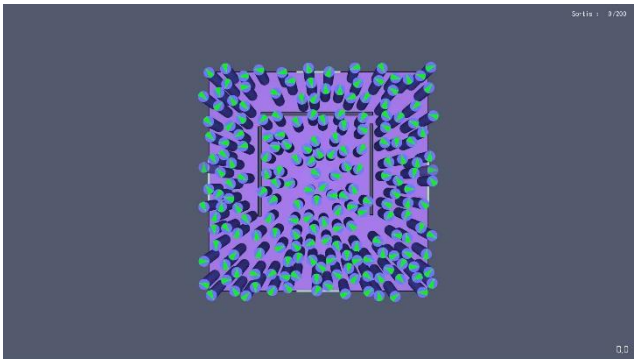


Temps : 20 | Sortis : 193/200

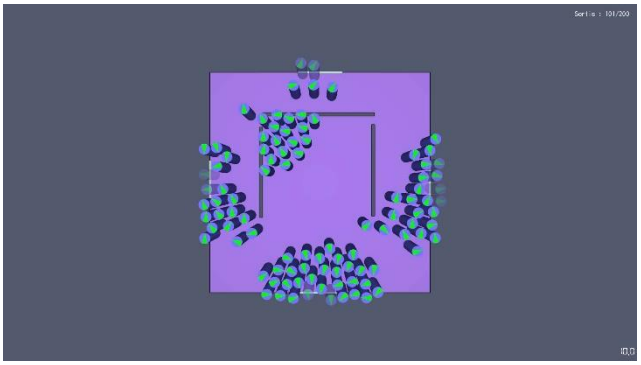


Temps : 28.4 | Sortis : 200/200

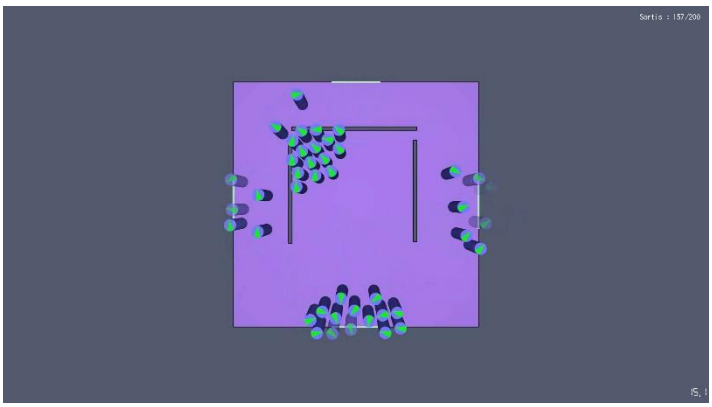
e. 4 sorties 3 murs



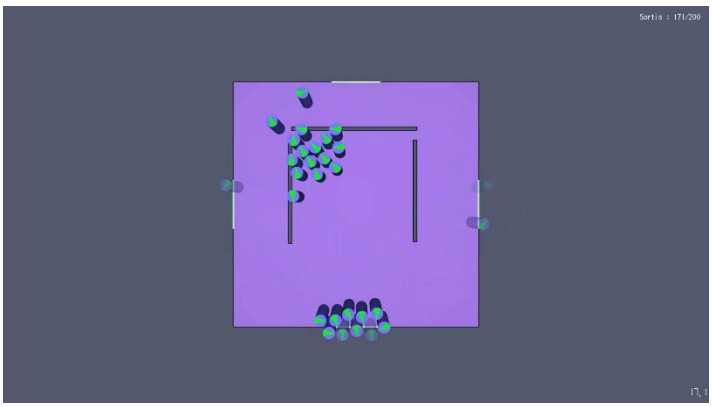
Temps : 0 | Sortis : 0/200



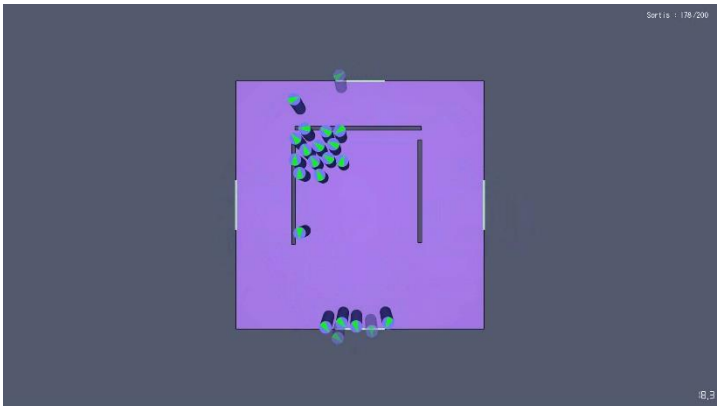
Temps : 10 | Sortis : 101/200



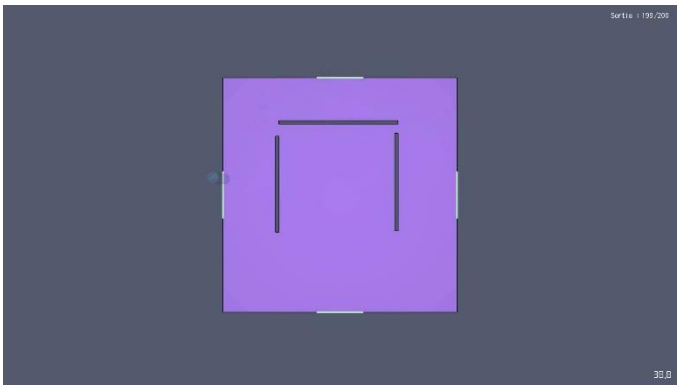
Temps : 15.1 | Sortis : 157/200



Temps : 17.1 | Sortis : 171/200

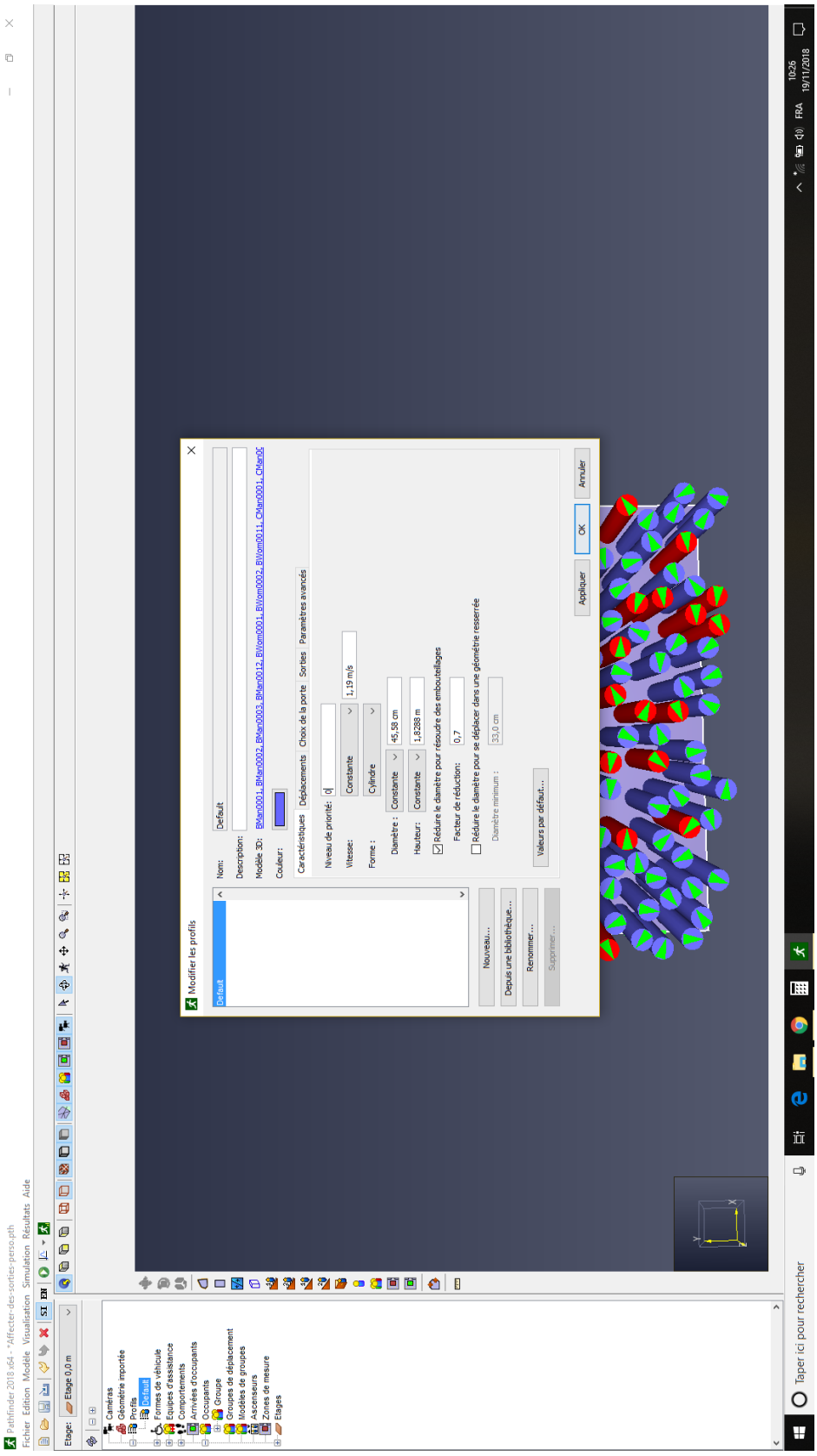


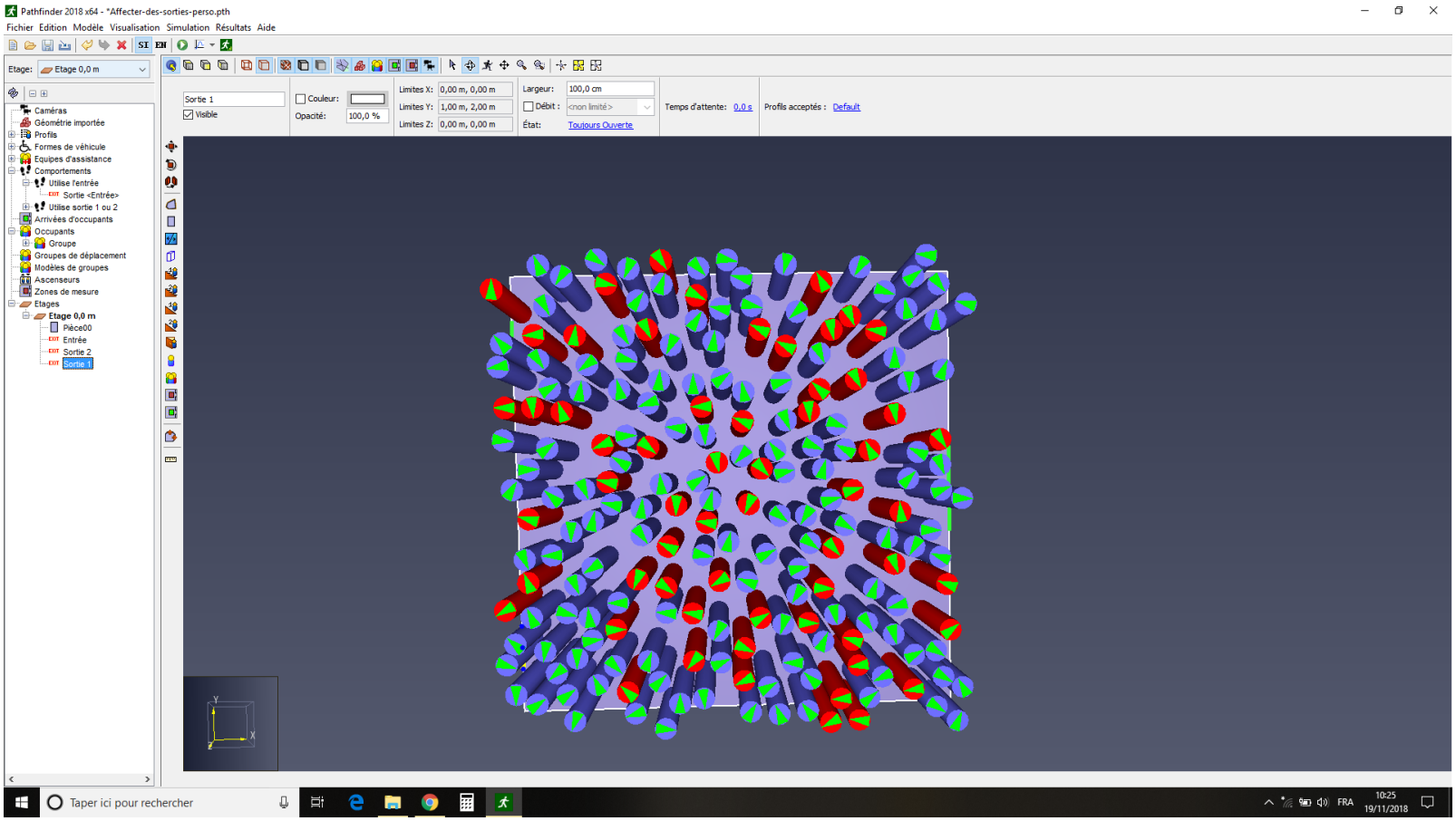
Temps : 18.3 | Sortis : 178/200

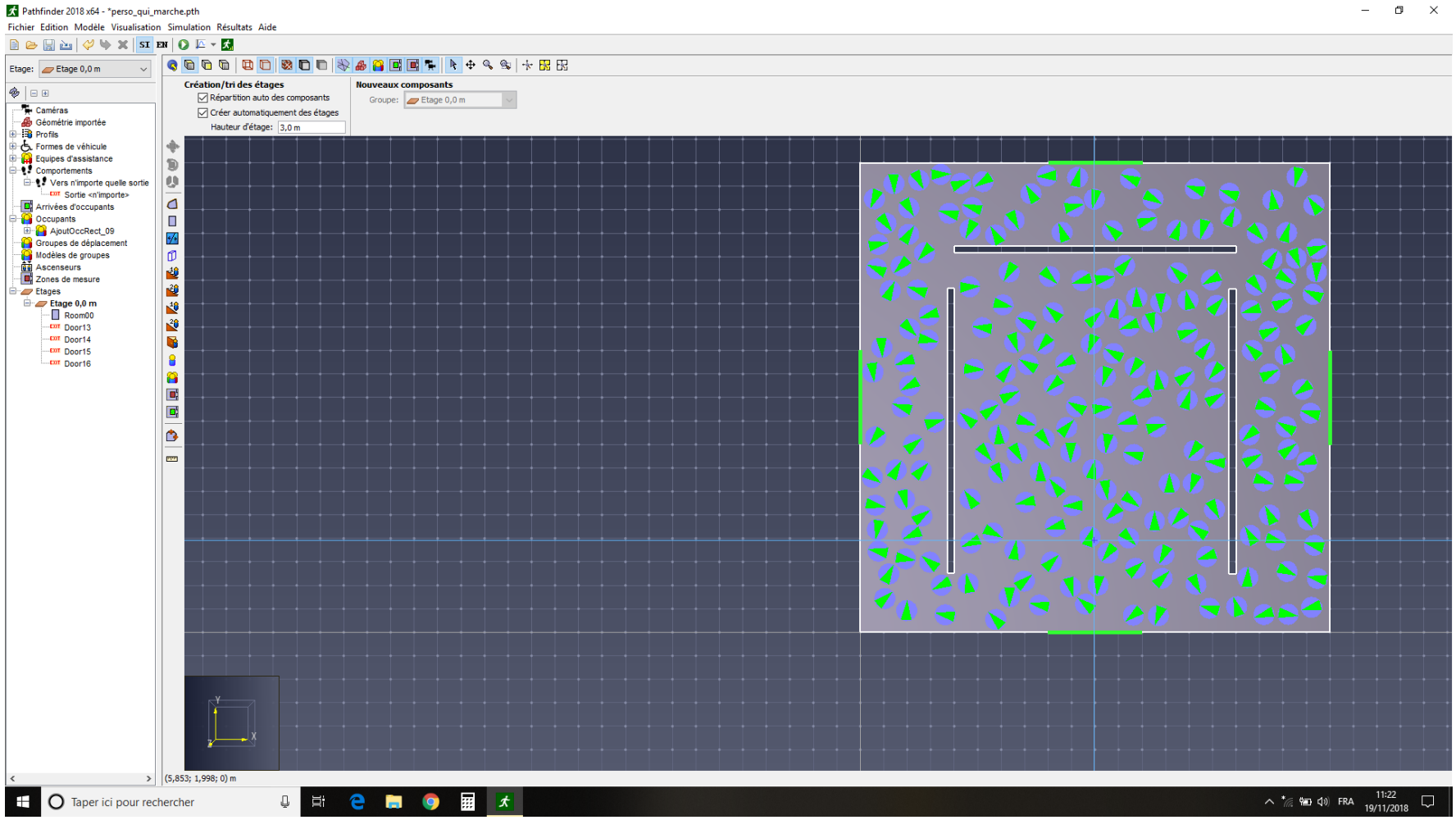


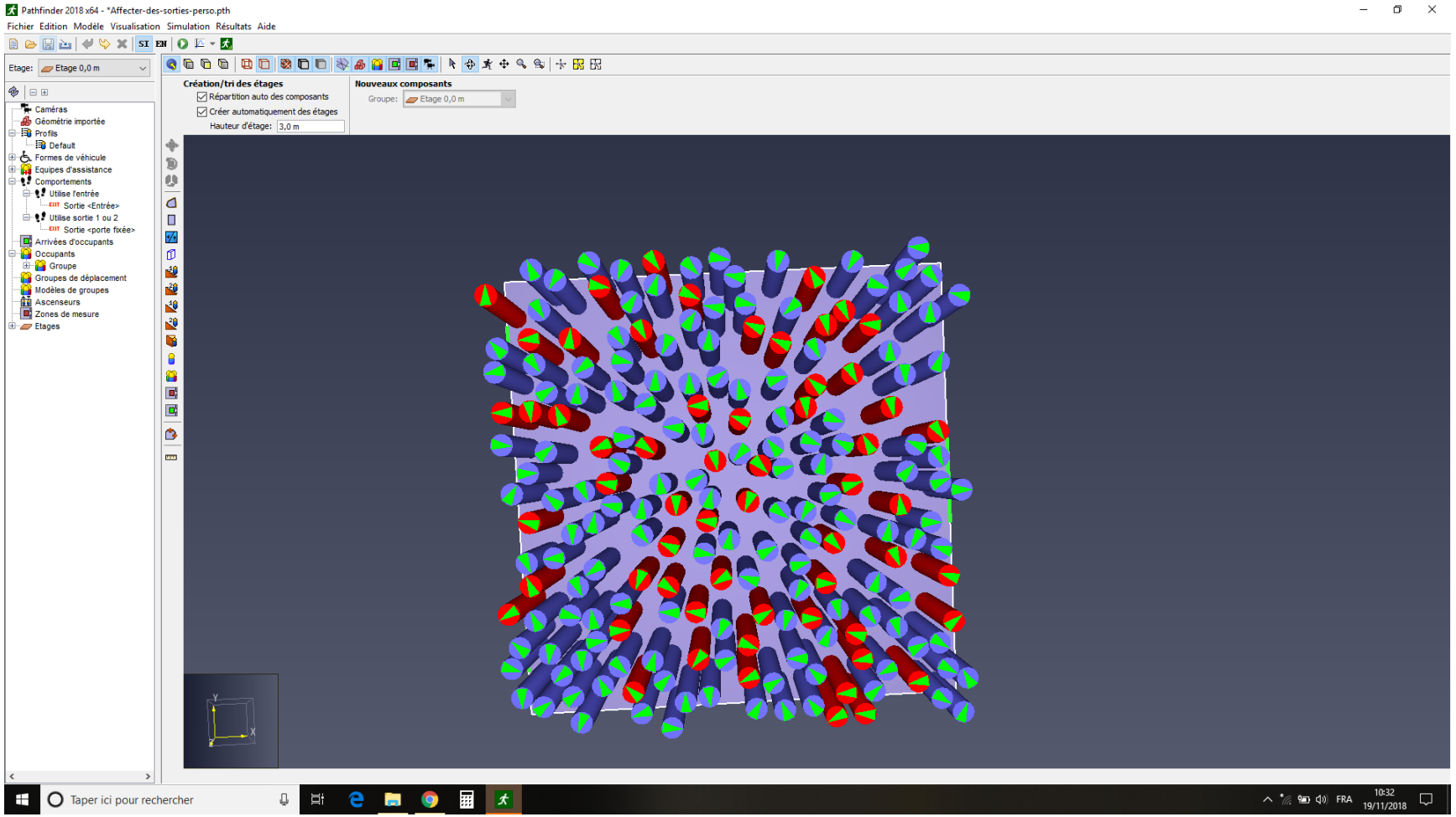
Temps : 38.8 | Sortis : 200/200

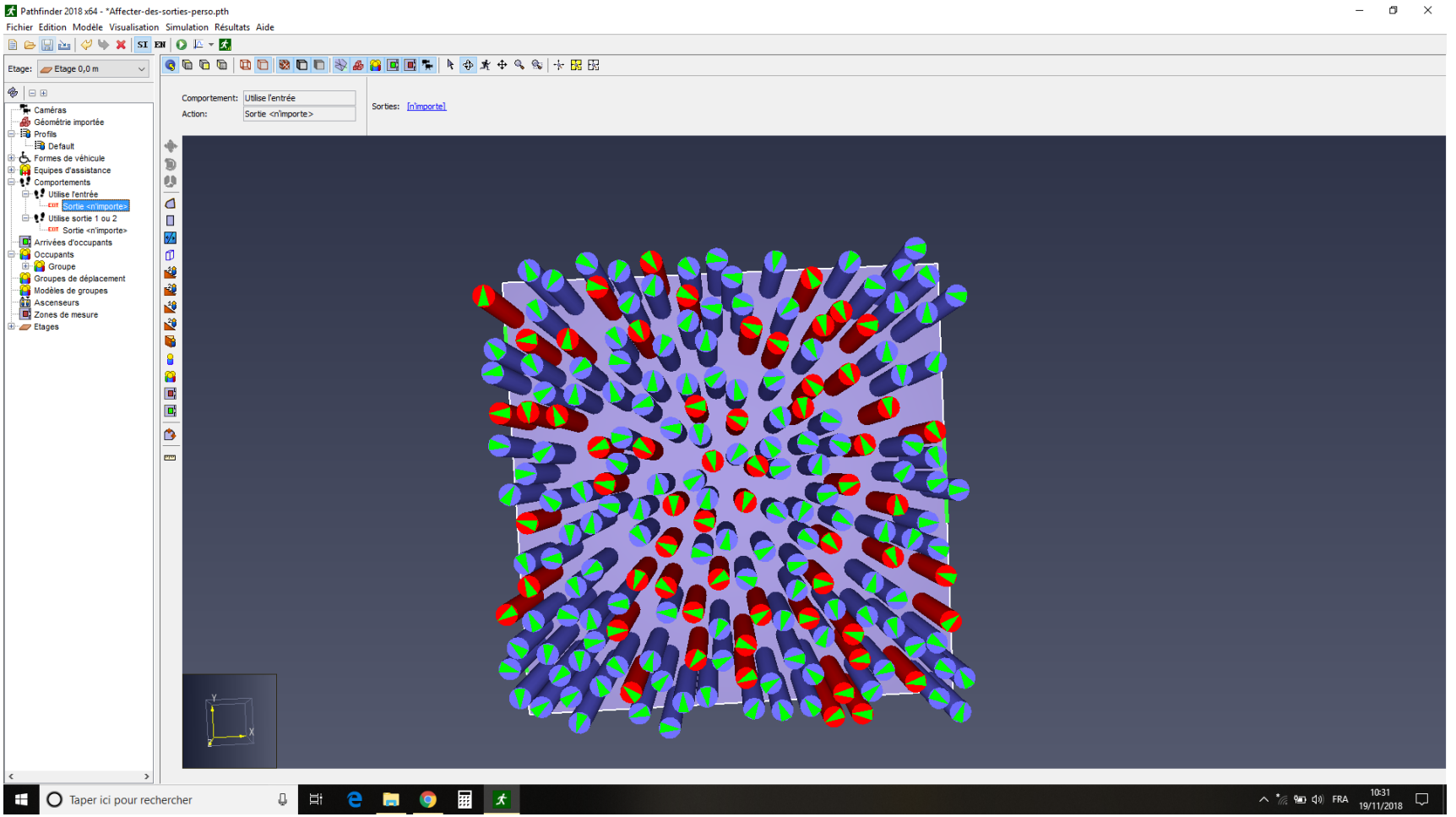
f. Paramètres




Pathfinder 2018 x64 - *Affecter-des-sorties-perso.pth
Fichier Edition Modèle Visualisation Simulation Résultats Aide
Etage: Etage 0,0 m
Caméras
Géométrie importée
Profils
Formes de véhicule
Equipes d'assistance
Comportements
Utilise l'entrée
Sortie <Entrée>
Utilise sortie 1 ou 2
Arrivées d'occupants
Occupants
Groupe
Groupes de déplacement
Modèles de groupes
Ascenseurs
Zones de mesure
Etages
Etage 0,0 m
Pièce00
Entrée
Sortie 2
Sortie 1
Sortie 1
Visible
Couleur:
Opacité: 100,0 %
Limites X: 0,00 m, 0,00 m
Limites Y: 1,00 m, 2,00 m
Limites Z: 0,00 m, 0,00 m
Largeur: 100,0 cm
Débit : <non limité>
État: Toujours Ouverte
Temps d'attente: 0,0 s
Profils acceptés : Default
Taper ici pour rechercher
FRA
10:25
19/11/2018


Pathfinder 2018 x64 - *perso_qui_marche.pth
Fichier Edition Modèle Visualisation Simulation Résultats Aide
Etage: Etage 0,0 m
Caméras
Géométrie importée
Profils
Formes de véhicule
Equipes d'assistance
Comportements
Vers n'importe quelle sortie
Sortie <n'importe>
Arrivées d'occupants
Occupants
AjoutOccRect_09
Groupes de déplacement
Modèles de groupes
Ascenseurs
Zones de mesure
Etages
Etage 0,0 m
Room00
Door13
Door14
Door15
Door16
Création/tri des étages
Répartition auto des composants
Créer automatiquement des étages
Hauteur d'étage: 3,0 m
Nouveaux composants
Groupe: Etage 0,0 m
(5,853; 1,998; 0) m
Taper ici pour rechercher
FRA
11:22
19/11/2018


Pathfinder 2018 x64 - *Affecter-des-sorties-perso.pth
Fichier Edition Modèle Visualisation Simulation Résultats Aide
Etage: Etage 0,0 m
Caméras
Géométrie importée
Profils
Default
Formes de véhicule
Equipes d'assistance
Comportements
Utilise l'entrée
Sortie <Entrée>
Utilise sortie 1 ou 2
Sortie <porte fixée>
Arrivées d'occupants
Occupants
Groupe
Groupes de déplacement
Modèles de groupes
Ascenseurs
Zones de mesure
Etages
Création/tri des étages
Répartition auto des composants
Créer automatiquement des étages
Hauteur d'étage: 3,0 m
Nouveaux composants
Groupe: Etage 0,0 m
Taper ici pour rechercher
FRA
10:32
19/11/2018


Pathfinder 2018 x64 - *Affecter-des-sorties-perso.pth
Fichier Edition Modèle Visualisation Simulation Résultats Aide
Etage: Etage 0,0 m
Caméras
Géométrie importée
Profils
Default
Formes de véhicule
Equipes d'assistance
Comportements
Utilise l'entrée
Sortie <n'importe>
Utilise sortie 1 ou 2
Sortie <n'importe>
Arrivées d'occupants
Occupants
Groupe
Groupes de déplacement
Modèles de groupes
Ascenseurs
Zones de mesure
Etages
Comportement: Utilise l'entrée
Action: Sortie <n'importe>
Sorties: [n'importe]
Taper ici pour rechercher
FRA
10:31
19/11/2018